Struts Dialogs: DialogAction

1. Overview

DialogAction is an abstract **Action**, which is used as a web resource "code-behind" class. It is based on SelectAction and adds the following functionality:

- handles initialization event;
- buffers error messages;
- renders view corresponding to resource state;
- stores state (in an action form);
- separates input and output processes (using POST-redirect-GET pattern)

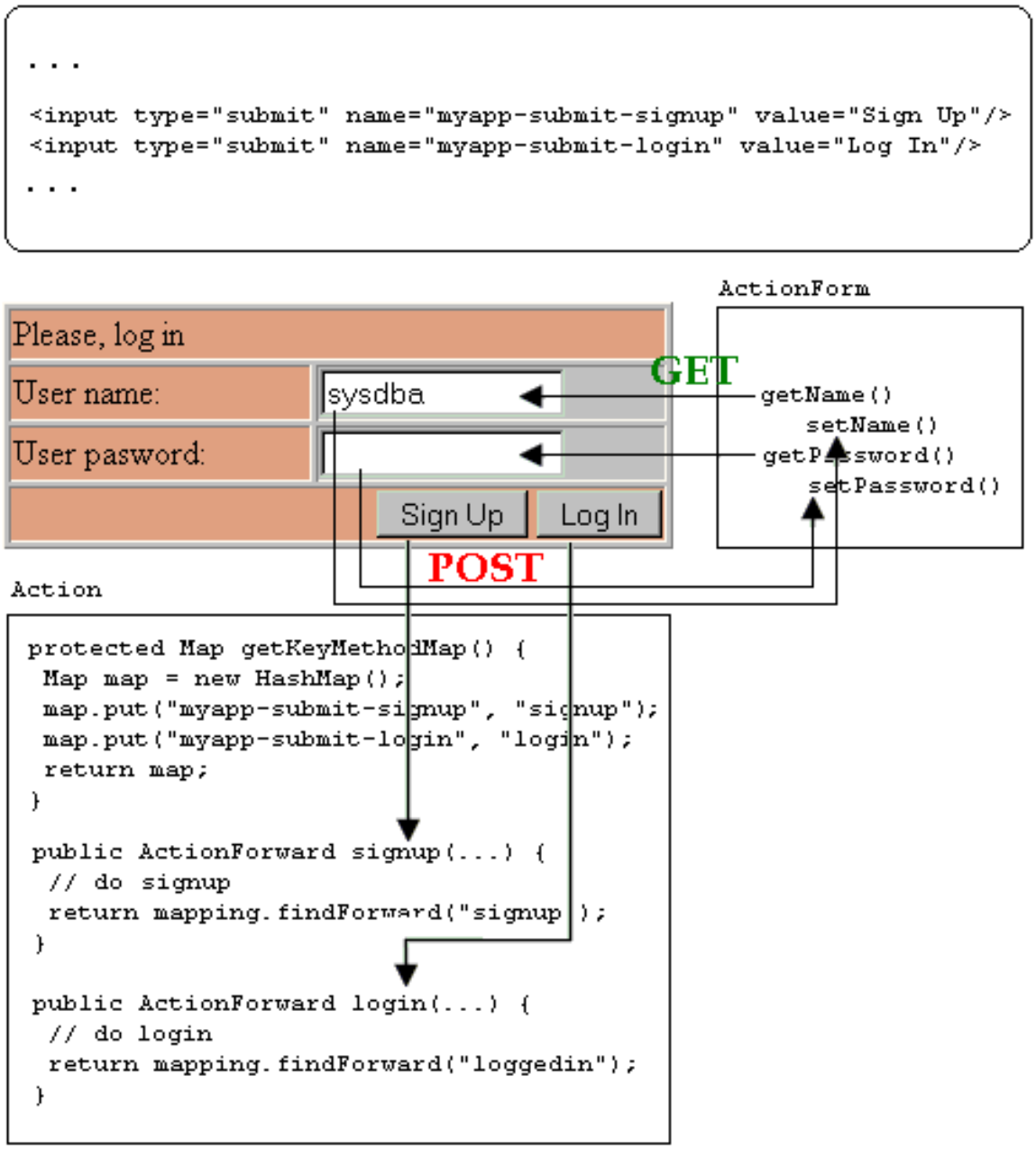
DialogAction allows to create robust user inteface, which can handle Refresh, Back and Forward buttons in user-friendly manner, and does not cause double submit issues or annoying POSTDATA messages.

2. Two-phase input processing

The most important concept of DialogAction is separation of input and output processes into two phases. The two-phase approach solves several issues, related to submission of HTML forms, like double submit problem, annoying POSTDATA messages and unfriendly user experience for Reload, Back and Forward buttons.

DialogAction has three different modes of operation:

- it sets up web resourse when it receives *initialization event*;
- it accepts input data, updates domain model and generates error messages on *input* phase
- it generates a result page during *render* phase.



DialogAction

HTTP specification recommends to use POST method to modify server data, while GET method should be used for requests with no side effects. Two-phase request processing nicely

corresponds to HTTP specifications, separating input and output phases by request method:

- browser submits input data using POST request;
- DialogAction accepts input and redirects browser to the location of result page;
- browser loads result page using GET request.

Because this pattern includes redirection, it can also be called as "Redirect-After-Post" or "Post/Redirect/Get".

It must be understood, that redirection is an inherent part of this pattern. Redirection allows to split one solid "POST input data, respond with result page" process into two phases, which are completely independent from server standpoint.

Separation of input and output improves usability and user experience:

- Any page can be reloaded without sending POST request to the server again.
- As a consequence, an application does not have to cope with implicit double submits.
- Another great usability consequence is that a user does not see "Do you want to resend POSTDATA?" message.
- Using Back and Forward buttons is safe, because only result pages are redisplayed. This behavior is defined in HTTP specification, which states that a browser must "forget" the POST request that preceds redirection response.

3. Usage

When an action is loaded either with hyperlink or by typing its location in the browser address field, browser sends GET request to the server. When DialogAction receives GET request, which does not contain event parameter, it renders a page. In case of simple dialogs, this would be default view. In case of multi-page resource this would be a page, corresponding to current resource state. The page usually contains HTML form, which accepts user data.

When form is submitted, DialogAction dispatches browser event to a corresponding handler method. To make this happen, automatic validation must be turned off in a corresponding form bean. The hander should validate input data explicitly, and perform model update if needed.

In case of error, the handler saves errors in the session and redirects to the same action. This is called *resource reloading*. It does not cause infinite loop, because request for data submission has POST type, while redirected request for action reloading is automatically converted to GET type. GET request does not cause reloading. Instead, when DialogAction receives GET request, it renders a view, corresponding to current state. The lifecycle repeats until the resource decides to hand control over to another resource, usually when user input does not contain errors.

```
<action path="/dialogloginaction"
  ...
  <forward name = "DIALOG-RELOAD" path="/dialogloginaction.do" redirect="true"/>
  <forward name = "DIALOG-VIEW" path="/dialogloginaction.jsp"/>
</action>
```

If no errors were detected and action can render a view for new resource state, then resource can hand control over to another action. Redirection is a preferred way to hand over the control to prevent double submit issues. DialogAction has default mapping name, used for resource reloading: "DIALOG-RELOAD".

When a user refreshes a page, DialogAction reloads itself, and displays a view appropriate to its state. This implies that application state must be stored on server between requests. The view is rendered by `getView` method. For a simple web dialog, which has only one view, the default implementation of getView is sufficient. All you need is define a name of JSP page in the `struts-config.xml` file, which corresponds to default "DIALOG-VIEW" mapping. See example above.

Do not use redirection for a view. If you create more complex web resource with several views, you need to override `getView` method and return a proper mapping for each resource state.

DialogAction has an option to reset the resource when it receives new portion of data submitted with POST request. POST request method signals that browser submits new data instead of loading existing information. But sometimes resource has to be reset with GET method, if it is navigated using hyperlink. DialogAction needs to distinguish these situations:

- reset resource and fill it with new data;
- navigate to resource using a "clean" link;
- refresh a resource explicitly;
- automatically reload a resource as the second phase of input processing.

DialogAction uses event parameter as flag for resource that it can be reset. Three other situations: "clean" linking, refreshing and resource reloading are treated the same, because request does not contain event parameter in all three cases. In these cases DialogAction renders a view, corresponding to current resource state.

In plain words the above means that navigating to a resource via link, which does not have event key as query parameter, displays current state of resource but does not initialize it. See live demos for details, and pay attention to resource URLs.

4. Notes

- Make sure that you turned "validate" property off in your action mapping in struts-config.xml file. Errors must be processed explicitly in the action class.

- Make sure that your action form uses session scope to retain form values between requests. If you are strongly against storing application state in the session, Struts Dialogs library will not suit you.

5. Sample Code

The following examples show how DialogAction can be used:

- Single-state Dialog
- Multi-state Dialog